PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTERY OF HIGHER EDUCATION AND SCIENTIFIC RESARCH

University Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculty of Sciences & technology

Department of Electronics

Report

End of Cycle Project (PFC)

MCIL 3

FIELD : **ELECTRONICS**

Speciality : Electrical industrial

By

ZIOUAR Hamza

SAIDI Adel

Entitled

Implementation of basic image processing algorithms using Xilinx System Generator

Presented on : 26/06/2022

Before the Jury composed of :

Name & surname	*Grade*	*Quality*	*Establishment*
M.YOUSFI. A	*Dr*	*President*	*Univ-BBA*
M. DJELLAL D	*MAA*	*Examiner*	*Univ-BBA*
Mrs. HAMADACHE Fouzia	*MAA*	*Supervisor*	*Univ-BBA*

Academic Year 2021/2022

Abstract

This report focuses on the implementation of basic image processing algorithms on FPGA using the powerful Xilinx System Generator (XSG) tool. We have modeled and simulated the HDL model of various image-processing algorithms such as color to grayscale and color image negative, enhancement, contrast stretching and image thresholding using XSG. The models have been implemented on FPGA by a compilation for hardware Co-Simulation.

Keywords: Image-processing algorithms, FPGA, XSG, Hardware Co-simulation.

ملخص

تقدم هذا المذكرة طريقة معالجة الصور باستخدام مولد نظام اكسلينيكس. يحتوي مولد نظام اكسلينيكس على مكتبات ضرورية لمساعدة أنواع مختلفة من الخوارزميات. تم دمجه مع بيئة ماتلاب سيمولينك في هذا العمل. يتم استخدام نهج يركز تقريره على تنفيذ خوارزميات معالجة الصور الأساسية على FPGA باستخدام أداة Xilinx System Generator (XSG) القوية. لقد قمنا بنمذجة ومحاكاة نموذج HDL للعديد من خوارزميات معالجة الصور مثل اللون إلى التدرج الرمادي واللون للصور السلبية ، والتعزيز ، وتمدد التباين ، وعتبة الصورة باستخدام XSG. تم تنفيذ النماذج على FPGA من خلال تجميع لمحاكاة الأجهزة.

الكلمات الرئيسية: خوارزميات معالجة الصور ، XSG ، FPGA ، محاكاة الأجهزة.

Résumé

Le présent mémoire porte sur l'implémentation d'algorithmes de base du traitement d'images sur FPGA à l'aide du puissant outil Xilinx System Generator (XSG). Nous avons modélisé et simulé le modèle HDL de divers algorithmes de traitement d’images tels que conversion d'images couleur en niveaux de gris, amélioration d’images, seuillage, etc à l'aide de XSG. Le modèle a été implémenté sur FPGA par une compilation pour une Co-Simulation matérielle.

Mots-clés : algorithmes de traitement d’images, FPGA, XSG, Hardware Co-simulation..

ACKNOWLEDGMENTS

First of all, we would like to express our gratitude to almighty Allah for his blessings, strength and ability to understand, learn and write this research. This thesis becomes reality with the kind support and help of many individuals.

We would like to express our sincere gratitude to our supervisor Md. Fouzia Hamadache, for her guidance, enthusiastic supervision, constant advice, and encouragement throughout the course of this research.

We are highly grateful to all our teachers for their patience and knowledgeable lectures. And my mom “aicha berrani” hamza and “yassine saadi” Special thanks go to the jury members and “Moussa MCIL5”for their precious time reading and analyzing our work. Our thanks and appreciations also go to our colleagues and people who have helped us out with their abilities.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 : Basic Image Processing Algorithms.

Chapter 2 : Image processing algorithms using xilinx systeme generator (XSG)

INTRODUCTION

Research in the field of image processing and analysis has been evolving rapidly over the last decade. Nowadays, image processing has various applications in the fields of medical imaging, meteorology, computer vision, microscopy, etc. [1]. Image processing on FPGAs is complicated because of the need to use separate architectures for image processing. Matlab, Simulink and Xilinx System Generator tools, which convert the image into appropriate formats supported by the FPGA, are used to facilitate these operations. XSG provides a powerful tool for designing, testing and implementing models on embedded systems [2].

To move from a model built with Simulink to a hardware model, System Generator offers a hardware/software co-simulation that allows to build a hardware version of the model and to use the Simulink simulation environment to validate the system functionality in hardware.

The present work consists in implementing, on FPGA, the basic image processing algorithms using the powerful Xilinx System Generator (XSG) tool under Simulink.

This thesis is organized as follows:

The first chapter details the basic image processing algorithms, image enhancement, thresholding, contrast stretching, etc. in order to model them under Simulink.

The second chapter is devoted to the modeling of the image processing algorithms using the XSG tool, to the simulation of the HDL model under the Simulink environment and finally to its implementation on FPGA using a compilation for a hardware Co-Simulation.

Finally, as a general conclusion, we synthesize the work and we identify the avenues of reflection.

Chapter 1

Basic Image Processing Algorithms

1.1 Introduction

Image processing can be defined as the set of methods and strategies operating on the image in order to extract the most relevant information or simply promote to provide a picture in addition to noticeable to the human eye.

In this chapter we present some basic thoughts of the field of digital image processing such as: image definition, image types, image characteristics, image processing system, elementary algorithms and in blade some concrete examples of image processing.

I.2 Definition and representation of an image

The definition of the term "image" itself, as given by the Petit Robert, encompasses a multitude of distinct meanings. It goes from the "exact reproduction or analogical representation of a being, of a thing", to the "mental representation of sensitive origin". The image is a visual representation of something (object, living being, concept...).[1].

A digital image is a table of colored points (pixels). For example, a basic computer screen displays 1024 pixels in width and 768 pixels in height.

The digital image is represented as a two-dimensional array containing integer values [1].

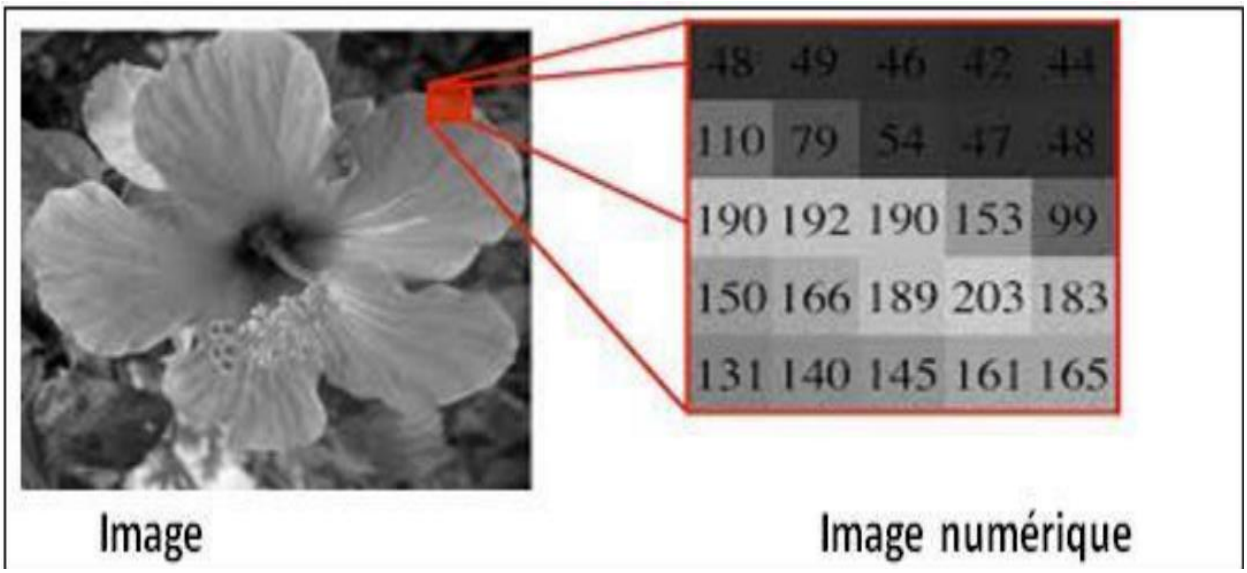


Figure 1.1 Image and digital image

1.3 Types of images

There are three types of images, which are as follows:

1.3.1 Color image

In a color image each pixel has a standard color described by the quantity of these 3 components: red (R), green (G) and blue (B), each of these colors is coded on the interval [0, 255]. The combination of these three colors gives a point of light (a pixel) of a certain color. So the RGB system is one way to describe a color in computing. Standard example: The trio {255, 255, 255} will give white, {255, 0, 0} a pure red, {100, 100, 100} a gray, and so forth. The first number gives the red component, the second the green component and the last the blue component [2].

The coding of the color is carried out on three bytes, each byte representing the value of a color component by an integer from 0 to 255. These three values generally code the color in the RGB space. The number of different colors which can be thus represented is 256 x 256 x 256 possibilities, that is to say nearly 16 million colors. As the difference of nuance between two very close but different colors in this mode of representation is almost imperceptible for the human eye, one considers conveniently that this system allows an exact restitution of the colors, this is why one speaks about "true colors".

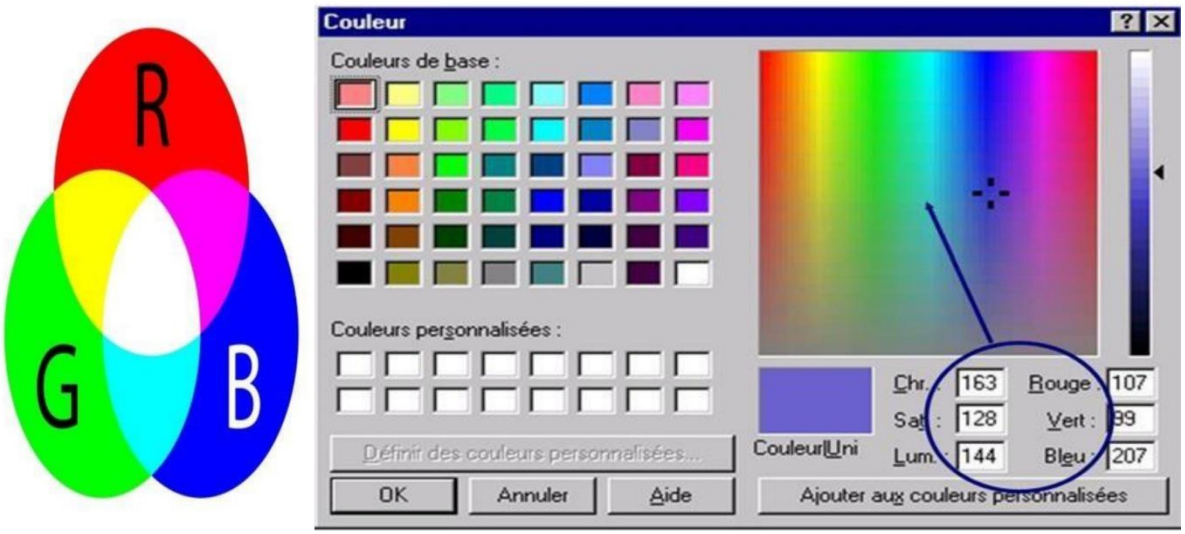


Figure I.2: color scheme

1.3.2 Grayscale image

One codes here only the level of the luminous intensity, generally on a byte (256 values). By convention, the value zero represents the black (null luminous intensity) and the value 255 the white (maximum luminous intensity).

This process is frequently used to reproduce photographs in black and white or text under certain conditions (with use of a filter to soften the contours in order to obtain smoother characters). The color of the image pixel has gray level can take values ranging from

black to white through a finite number of intermediate levels so the values between 0 and 255. (0 is black). [2]

Figure 1.3 Gradation of values from black to white

1.3.3 Binary image

A binary image is an image for which each pixel can only have a value of 0 or 1. The manipulation of such images is full of specialized tools and mathematical theories for several reasons:

The beginnings of digital image processing did not allow the processing of complex images (problem of computing time, available memory space and quality of output devices). Moreover, the first applications around 1950 were well adapted to this type of images.

Binary images are a simple context allowing a mathematical formalization of problems by tools such as topology. In the field of industrial vision (defect detection, quality control, measurement, ...) we often consider the binary image as a necessary step, usually following the segmentation phase. Binary image contains with only 2 possible values: each pixel has value that 0 for black or 1 for white [2].

0	0	0	0	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	0	0	0	0

Figure 1.4 Image black and white

1.4 Basic image processing algorithms

In images, we generally perform operations pixel by pixel: addition, subtraction, multiplication, division, linear combination, ... with those of the map using a linear homogeneous transformation (translation, homothety, rotation, projection).

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies.

Analogue and digital image processing are the two types of image processing methods employed. Hard copies, such as prints and photographs, can benefit from analog image processing. When applying these visual techniques, image analysts employ a variety of interpretation fundamentals. Digital image processing techniques allow for computer-assisted alteration of digital images. Pre-processing, augmentation, and presentation, as well as information extraction, are the three general processes that all sorts of data must go through when employing digital technology.

1.4.1 Conversion of color image to grayscale

Converting a color image to a grayscale image with important details is a difficult task. The contrast, sharpness, shading, and structure of color images might be lost when converted to grayscale. A novel algorithm is presented to preserve the contrast, sharpness, shading, and structure of color images. The new technique conducts an RGB approximation to convert a color image to grayscale, lowering and enhancing chrominance and brightness. The grayscale images produced by the algorithm in the studies demonstrate that the system keeps

important color attributes as contrast, sharpness, shadow, and color structure. [3]. Color images are transformed to grayscale based on the color of each pixel containing red (R), green (G), and blue (B) to reduce processing time [4],[5].

The RGB image is converted to a grayscale image according to the following equation:

$$Y=0.3*R + 0.59*G + 0.11*B \quad (1)$$

Figure 1.5 Result obtained from image conversion to grayscale.

1.4.2 Convert grayscale image to black and white image (thresholding)

Replaces all pixels in the input image that are brighter than the level with the value 1 (white) and all other pixels with the value 0 (black) to convert a grayscale image to a binary image [6]. For each pixel, the rule can be utilized to create a binary decision.[7]

$$F(x,y) = \{1 \quad if \quad f(x,y) > Txy\ 0 \qquad otherwise \quad (2)$$

Where Txy is the threshold assigned to location (x, y) in the image f(x,y)

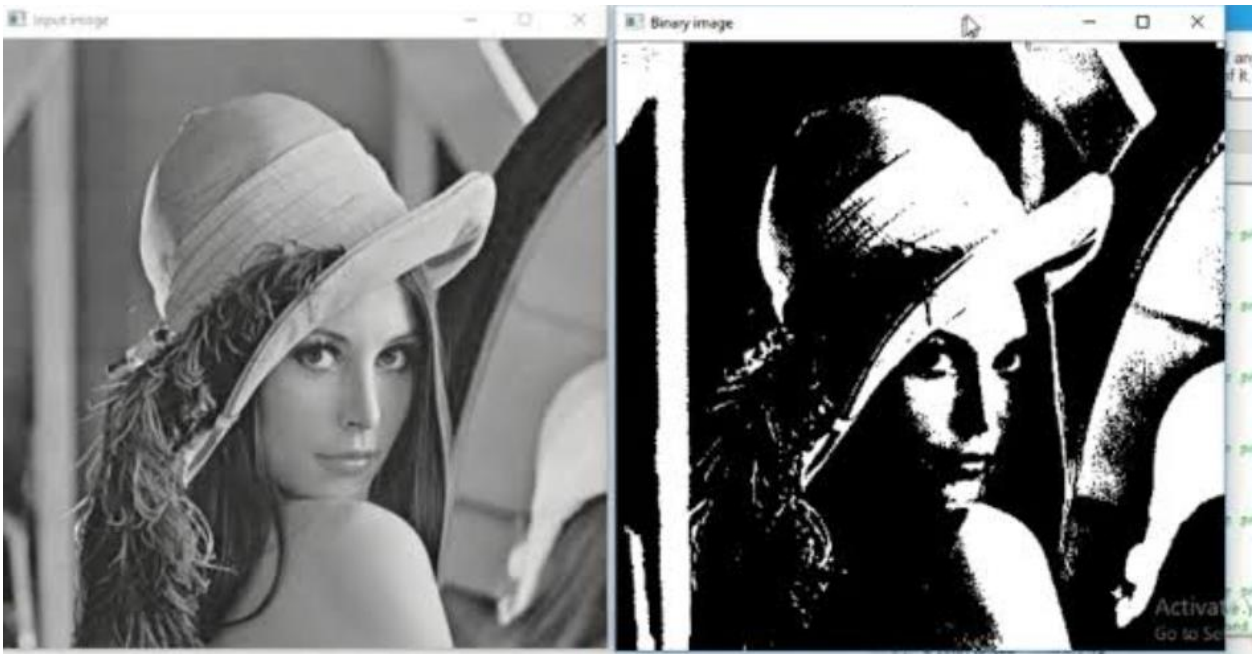


Figure 1.6 Result obtained from converting image to black and white.

1.4.3 Image negative

A negative image has light portions that appear dark and dark areas that appear light. Inverting a negative color image causes red parts to look as cyan, greens to appear as magenta, and blues to appear as yellow, and vice versa. Film negatives contain less contrast than final printed positive photos, but a larger dynamic range. When printed on photographic paper, the contrast usually rises. When scanning film negatives into the digital domain, the contrast can be altered at the time of scanning or, more broadly, during post-processing [7]

$$F(x,y) = f(x,y) - 255 \quad (3)$$

$$if\ F(x,y) < 0 \implies F(x,y) = 0 \quad (4)$$

$$if\ F(x,y) > 255 \implies F(x,y) = 255 \quad (5)$$

Figure 1.7 Result obtained from Grayscale Negative

1.4.4 Image enhancement

For human viewers, image enhancement increases the interpretability or perception of information in images. Images can be enhanced by adding some constant values to it. [7]

$$F(x,y) = f(x,y) + constant \quad (6)$$

Where constant is a number between 0 and 255

Figure 1.8 Result obtained from image enhancement

1.4.5 Algorithm for contrast stretching

The upper and lower pixel values for the image to be normalized must be specified before stretching may begin. Typically, these limits are simply the minimum and maximum pixel values permitted for the image type in question. For an 8-bit grayscale image, the lower and upper boundaries can be 0 and 255, respectively. Contrast stretching is the process of altering an image's contrast or luminosity characteristics.|7]. Grayscale image is stretched according to the equation.

$$New\_pixel = 3(old\ pixel - 127) + 112\ . \quad (7)$$

New_pixel is its result after the transformation.Values below 0 are set to 0 and values about 255 are se t to 255.

Figure 1.9 Result obtained from contrast stretching

1.5 Conclusion

In this chapter we have detailed the basic image processing algorithms to model them in Simulink in the next chapter.

Chapter 2

Image processing algorithms using Xilinx System Generator (XSG)

2.1 Introduction

Image processing is used to transform images in order to improve their quality and extract useful information. The need to process the image in real time, which leads to the implementation at the hardware level, which offers parallelism, and therefore significantly reduces processing time. Xilinx System Generator, a modeling and design tool that provides all the tools for easy graphical simulation in Simulink [8].

In this chapter, we focus on the implementation of various image processing algorithms such as negative generation, image enhancement, contrast stretching, image thresholding using the Xilinx System Generation (XSG) tool and transferring the design to the FPGA through Hardware Co-Simulation.

2.2 Xilinx System Generator

Xilinx System Generator is a Xilinx DSP configuration instrument that allows to use the model-based MATLAB/Simulink plan for execution in FPGA [9]. There are many DSP blocks that are adders, multipliers, registers, error revision blocks, FFTs, channels and memories in the Xilinx DSP blockset for Simulink. The general system DSP block graph is fully switched to RTL [10].

Xilinx System Generator (XSG) is an FPGA integrator design environment (IDE) that leverages Simulink as a development environment and presents as a block set. It features an integrated design flow that allows you to jump right to the configuration file (*.bit) needed to program the FPGA. This tool also allows hardware co-simulation used to test user-created cores on the target hardware simultaneously with the model present in the Simulink environment [11].

2.3 Image processing algorithms with xilinx system generator (XSG)

The XSG library includes a series of essential function blocks, typically used in FPGA target programming. The following figure illustrates the model of image processing algorithms developed with XSG.

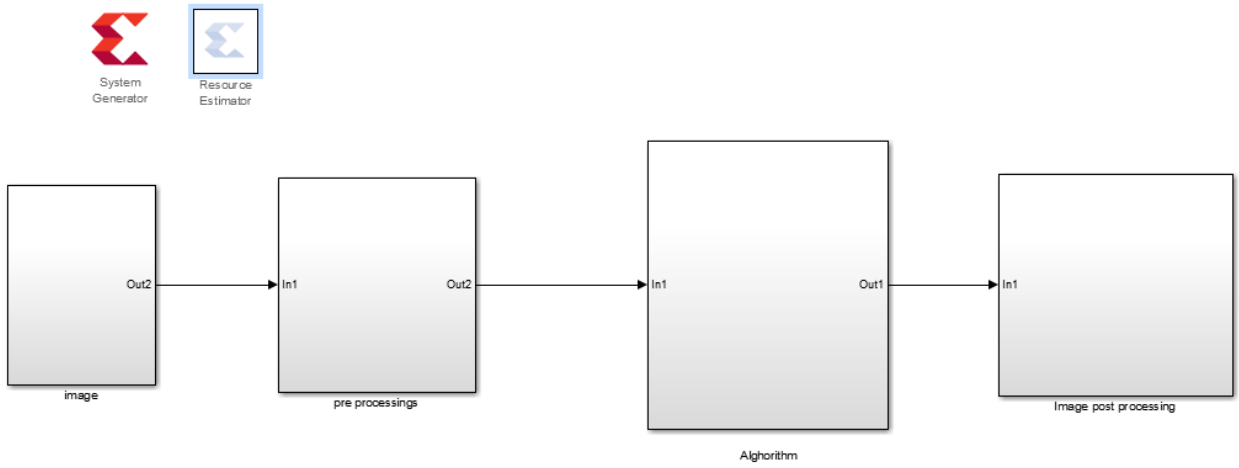


Figure. 2.1. Image processing algorithms using XSG

The entire operation for any image processing technique using Simulink and Xilinx blocks mainly goes through four phases such as image, preprocessing, algorithms and post processing.

2.4 Main blocks

The essential blocks in all algorithm designs in XSG are as follows:

2.4.1 System generator token

The System Generator token serves as a control panel for the simulation system and parameters. It's also used to start the netlisting code generation. At least one System Generator token must be present in every Simulink model that contains a Xilinx blockset element. at least one token from the System Generator It is possible to describe how code generation and simulation should be handled once a System Generator token has been added to a model [12]. The symbol and dialog box of the "System Generator" block are shown in the following figure:

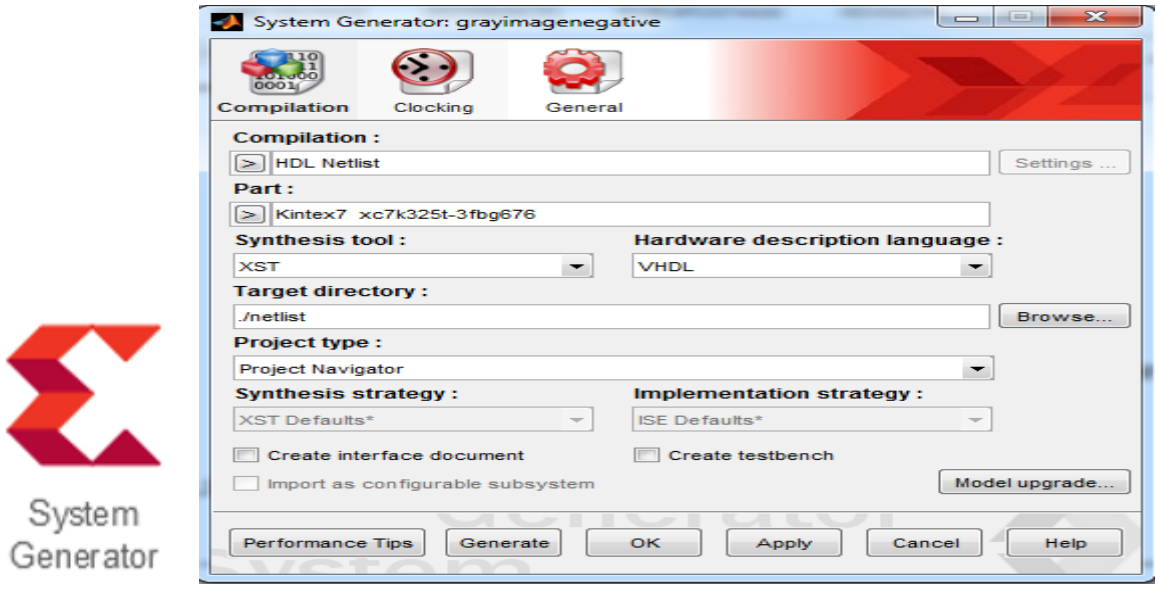


Figure. 2.2. Symbol and dialog box of the System Generator token.

2.4.2 Blocs Gateway In & Gateway Out

The Gateway In from Xilinx is an input block in the Xilinx part of the Simulink design. This block converts Simulink data types (integers, doubles) to the System Generator fixed-point type in one hand and in another hand, the gateway Out block is the output from the Xilinx part of a Simulink design. This block converts the System Generator's fixed-point or floating-point data type to a Simulink data type [12].

Figure. 2.3. Gateway In &Gateway Out

2.4.3 Source image

Image matrix is delivered by the block 'Image From file' which allows to import the image for processing. We choose grayscale image “kids.tif ” and color image “ onion.png ”.

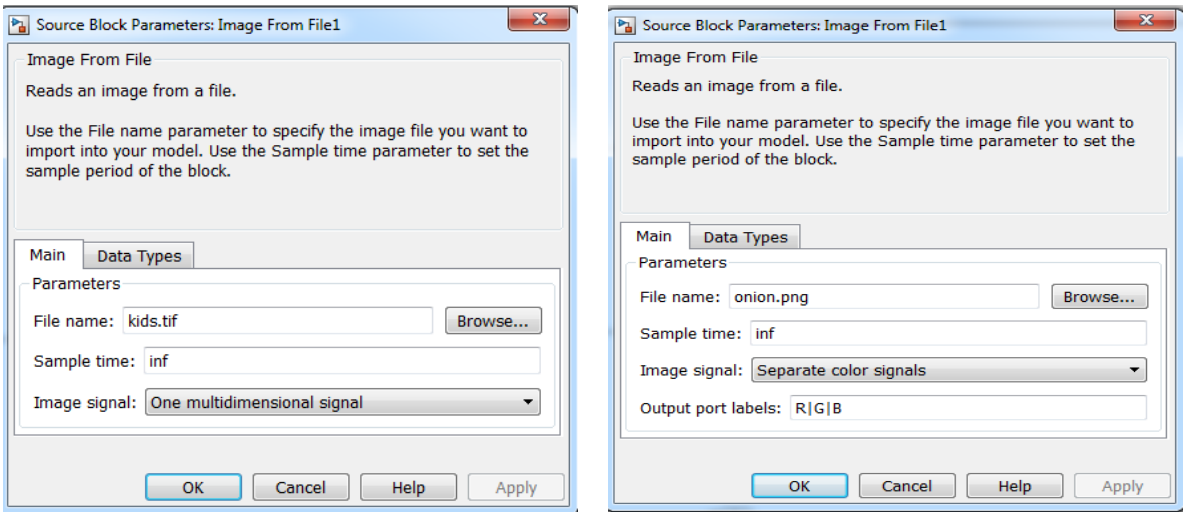


Figure. 2.4. Block parameters image source

The following are the images chosen for implementation using XSG.

Figure. 2.5. Images color and grayscale use in simulation

2.4.4 Image Pre-processing Blocks

Because the image is a two-dimensional (2D) layout, it must be preprocessed and supplied as a one-dimensional (1D) vector to match the hardware requirements. Preprocessing images in Matlab helps to provide data to FPGA as a specific test vector matrix suitable for FPGA Bitstream compilation using the system generator [13]. To process a 2D image, it is first transposed and then converted to 1D using the 2D to 1D conversion block. The frame conversion block maps the output signal into frame-based data and delivers it to the debuffing block, which then converts that frame into scalar samples at a higher sampling rate |7]. Figure 3.2 depicts the model-based design for image pre - processing stage

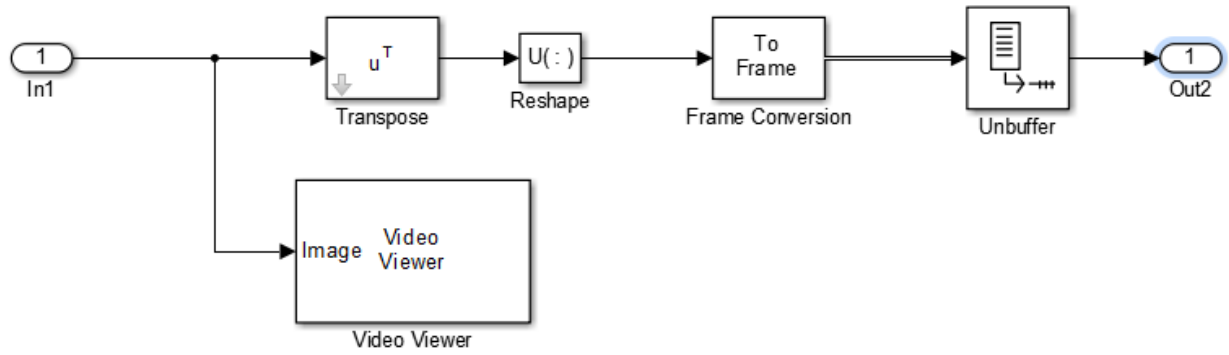


Figure. 2.6. Pre-Processing Block Diagram for grayscale image

We add matrix concatenate and Reshape blocks to achieve pre-processing of color image.

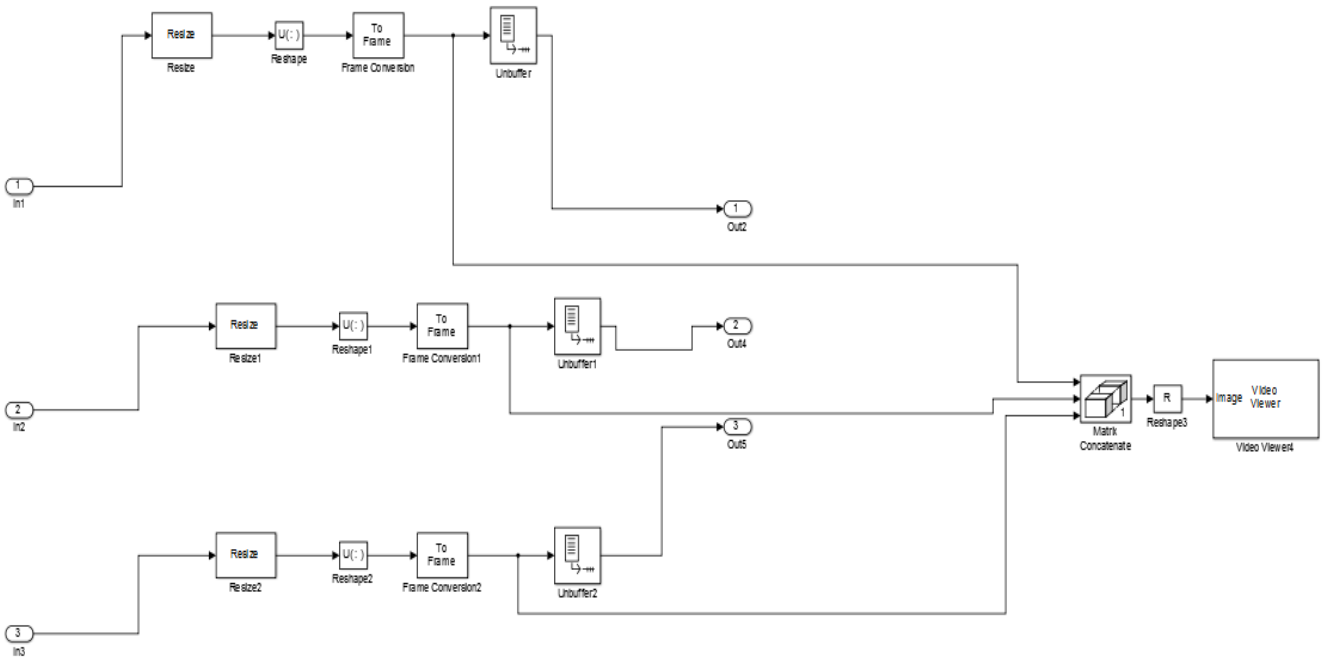


Figure. 2.7. Pre-Processing Block Diagram for color image

2.4.5 Image post processing blocks

Image post-processing is used to recreate the image from a 1-D vector. It consists of four blocks: data type conversion, buffer, convert 1-D to 2-D and transpose [12,13]. The first block encodes the image signal as an unsigned integer. In the second block, scalar samples are converted to frame output at a reduced sampling rate. The third block converts a one-dimensional image signal to a two-dimensional image matrix. The output image is transposed in the final block. The Image Post-processing processes are depicted in block

diagram in Figure 2.7.

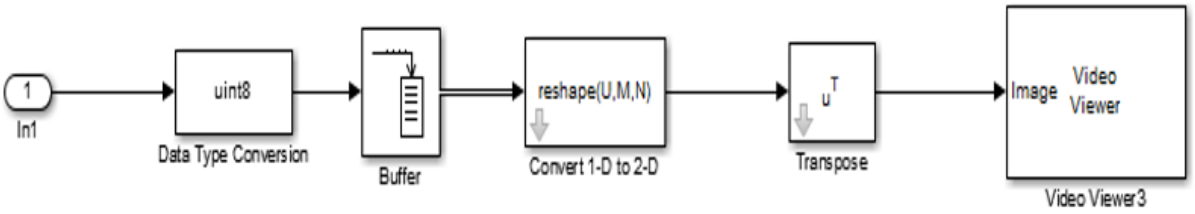


Figure. 2.8. Post-Processing Block Diagram for grayscale image

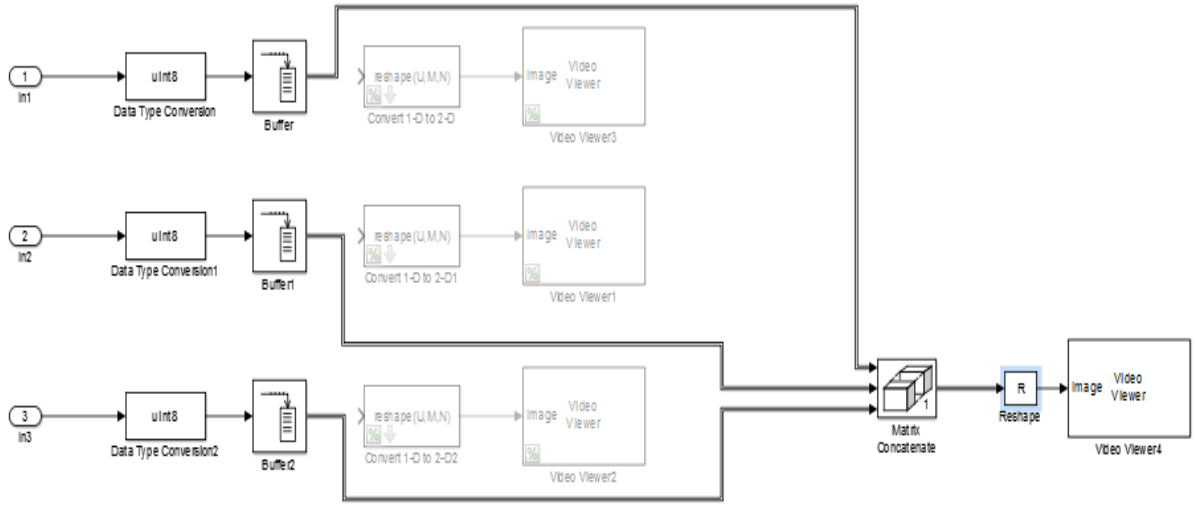


Figure. 2.9. Post-Processing Block Diagram for color image

2.5 Image processing algorithms

We aim to implement image processing algorithms using Xilinx System.

2.5.1 Algorithm for image negative

To construct the negation or inverse of an input image, the AddSub block directlly subtracts the constant 255 from the pixel value. Figure 2.10 shows the XSG design of Gray Scale Image Negative. The result of this algorithm is shown in Figure 2.11.

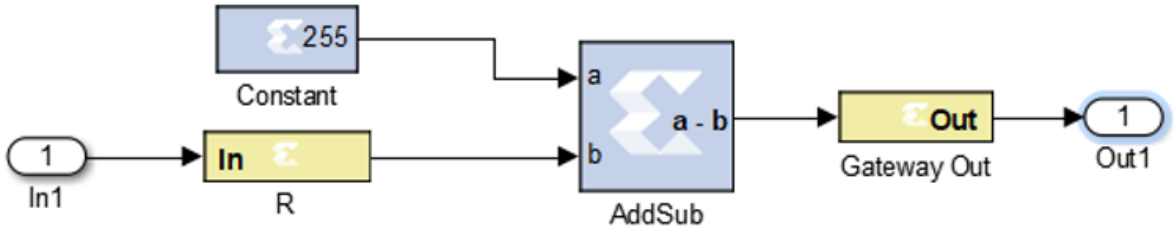


Figure. 2.10. XSG design of grayscale image negative

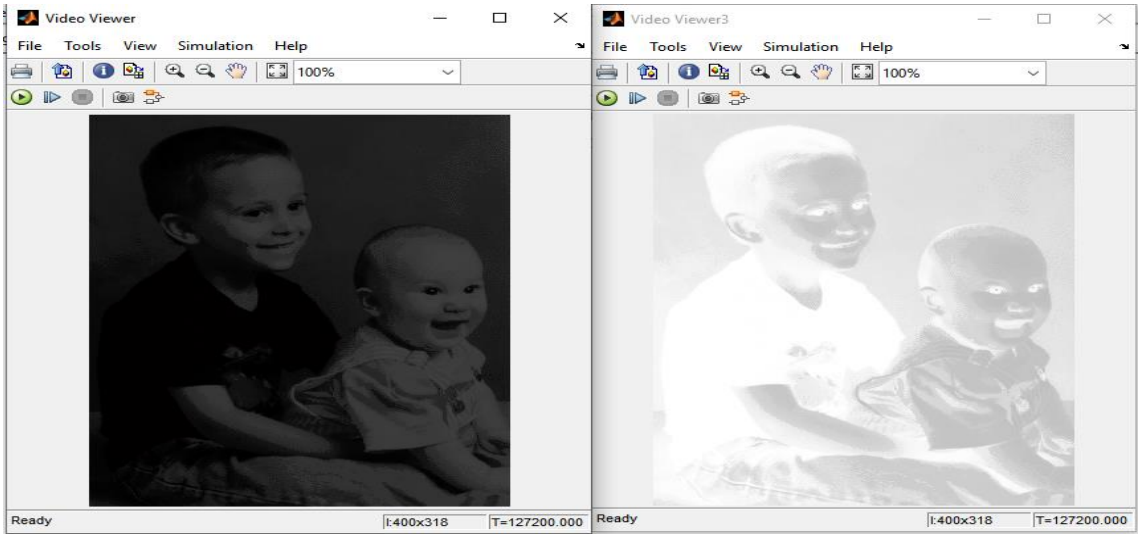


Figure. 2.11. Result for grayscale image negative

Figure 2.12 shows the XSG design of color Image Negative. The result of this algorithm is shown in Figure 2.13.

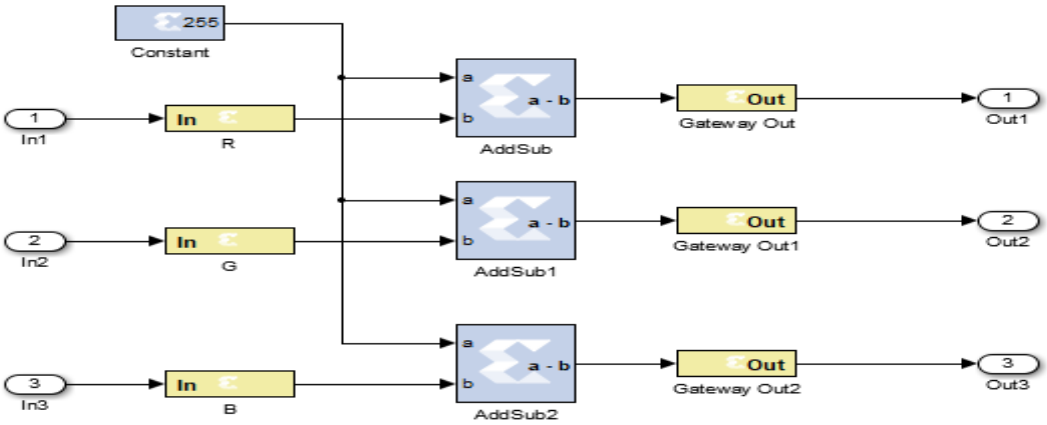


Figure. 2.12. XSG design of color image negative

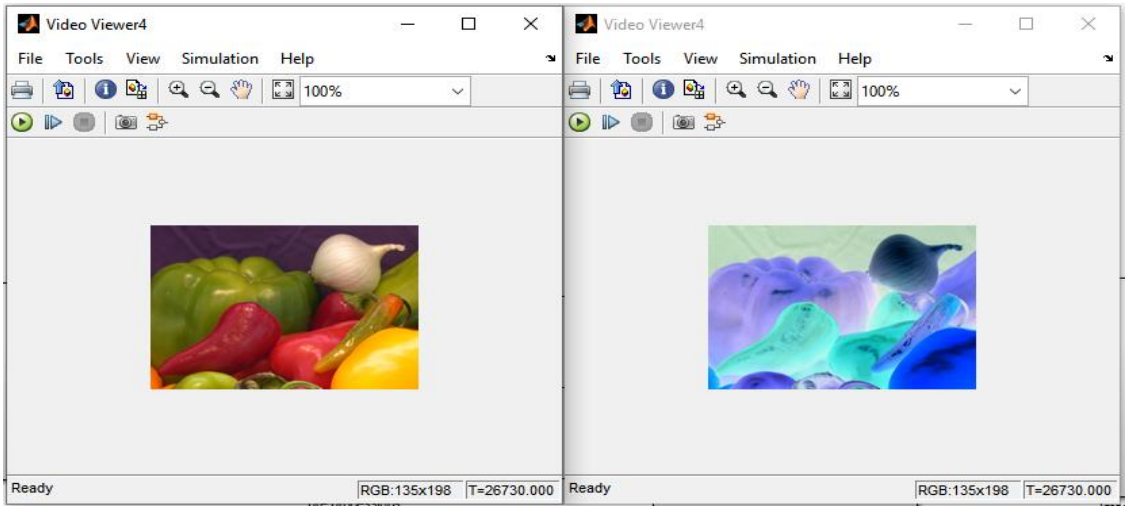


Figure. 2.13. Result for color image negative

2.5.2 Algorithm for contrast stretching

Contrast stretching is the process of altering an image's contrast or luminosity characteristics. The basic algorithm is described below. It can be achieved by employing a constant multiplier and AddSub blocks [8]. Figures 2.14 and 2.15 show the contrast stretching algorithm and results for gray scale image.

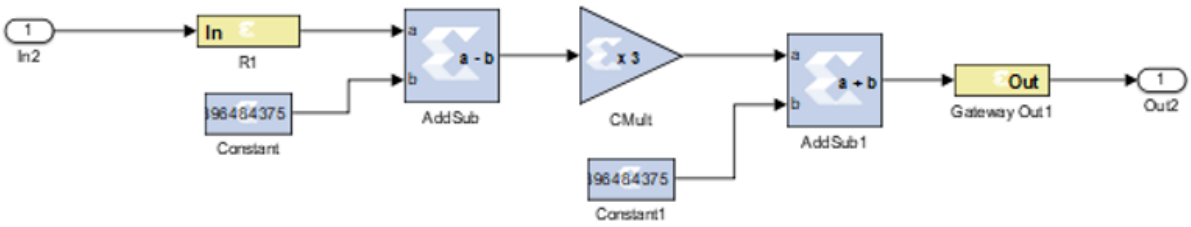


Figure. 2.14. XSG design of contrast stretching grayscale image

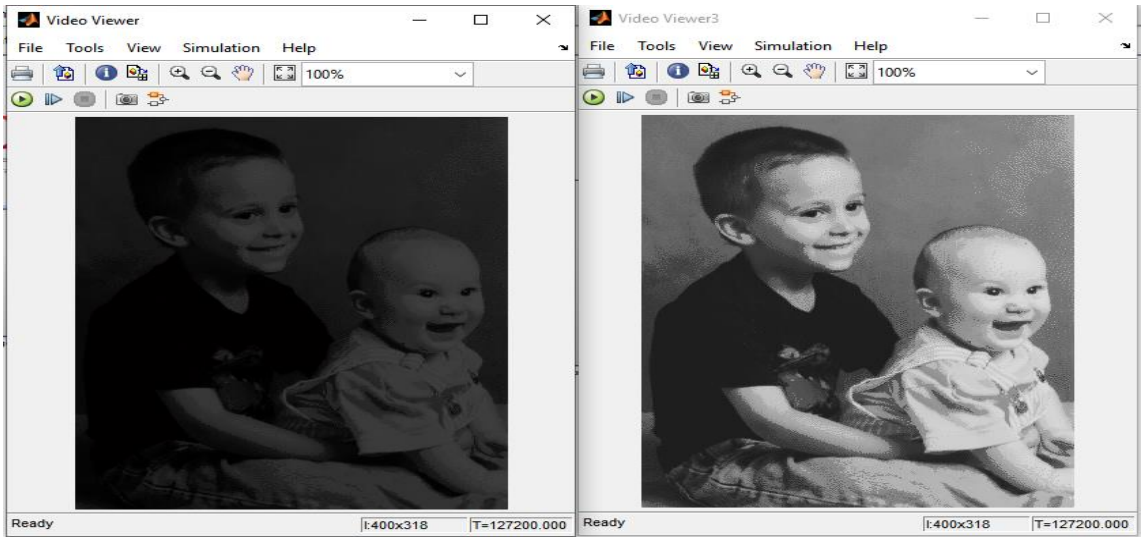


Figure. 2.15. Result for contrast stretching grayscale

Contrast stretching for color images stretches the intensity, tint, and saturation values for each pixel in an image for each R, G, and B signal. [6] Figures 2.16 and 2.17 show the contrast stretching algorithm and results for color image.

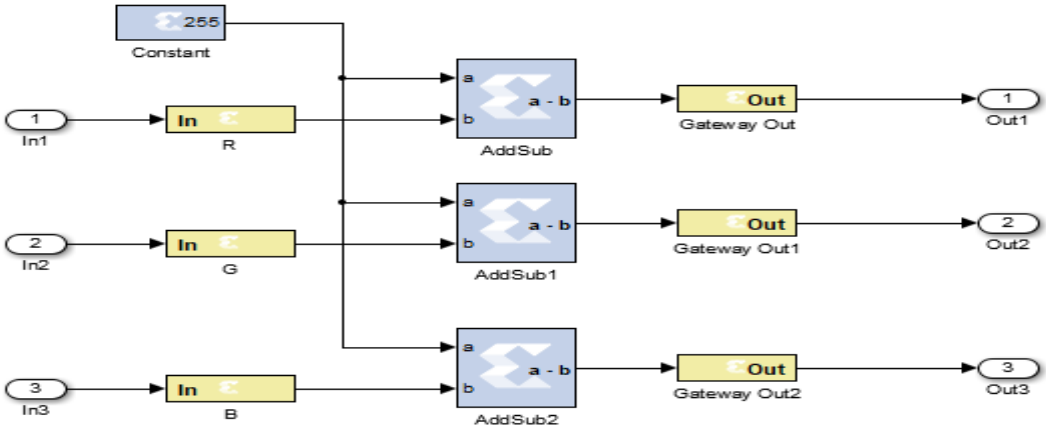


Figure. 2.16. XSG design of contrast stretching color image

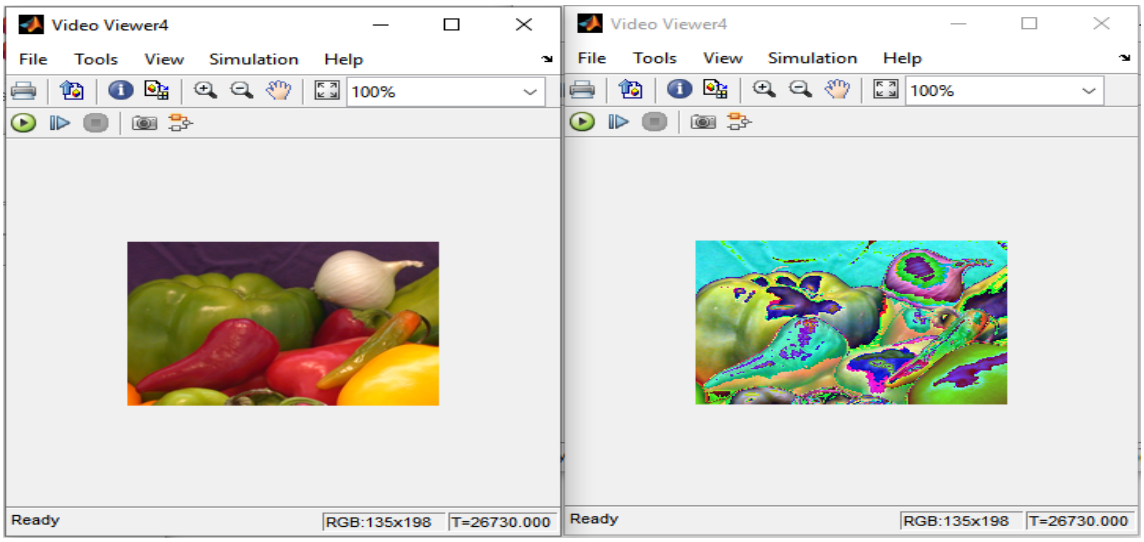


Figure. 2.17. Result for contrast stretching color image

2.5.3 Algorithm for thresholding

The process of making all pixels over a given level of white threshold while others are black is known as thresholding. A suitable constant 55, is utilized to implement the method, and a Mux is used to replace the pixel with white or black [7].

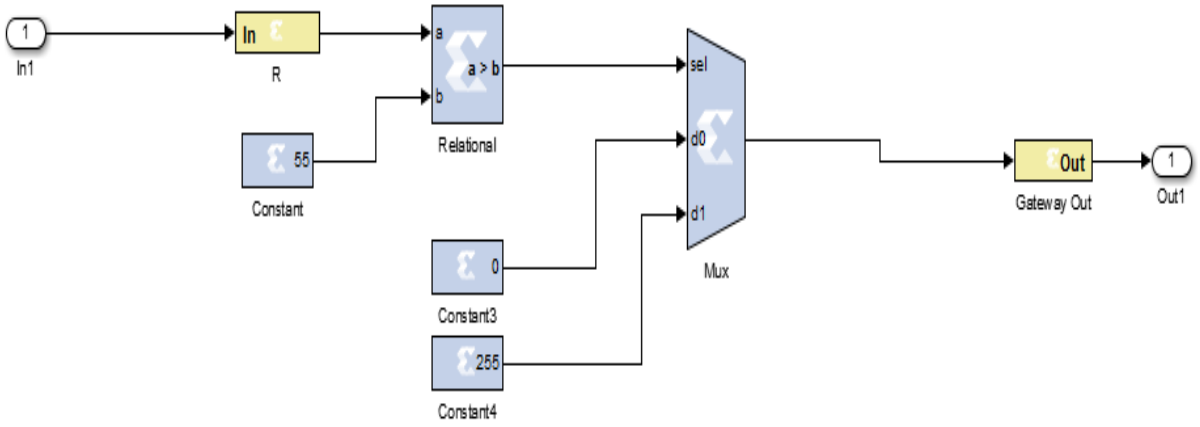


Figure. 2.18. XSG design of thresholding grayscale image

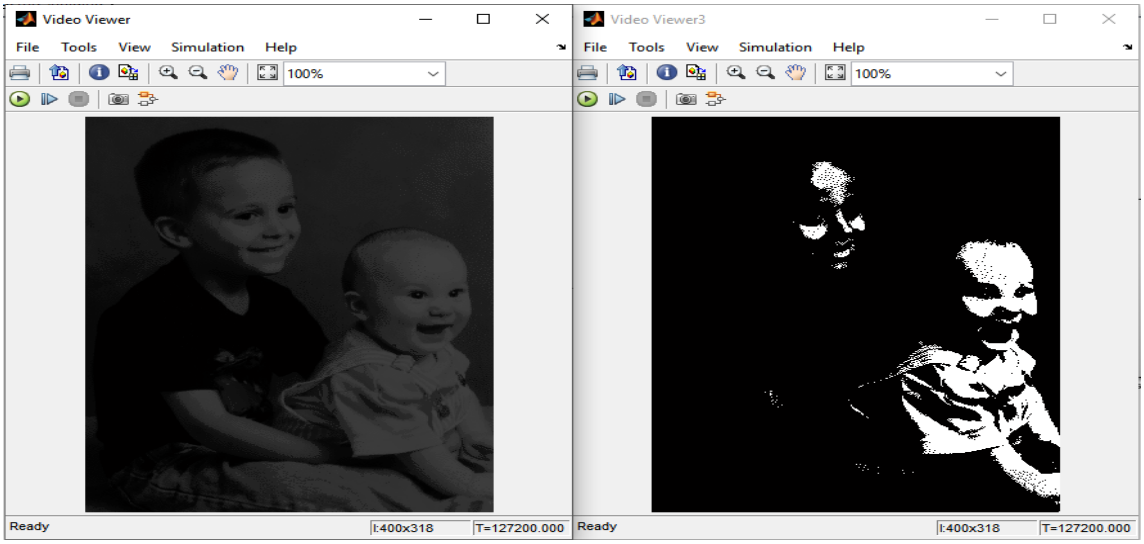


Figure. 2.19. Result for thresholding grayscale image

2.5.4 Algorithm for image enhancement

By merely adding a little illumination to some photographs, they can be improved for perception. The one-dimensional image signal is resolved for grayscale images, and the R|G|B multidimensional image signals are resolved similarly for color images [13].

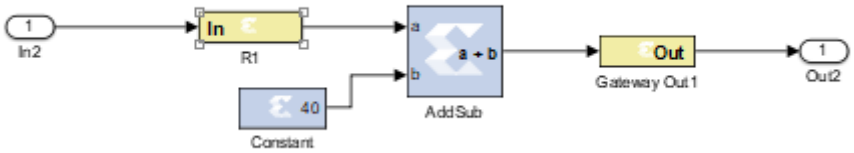


Figure. 2.20. XSG design of grayscale image enhancement

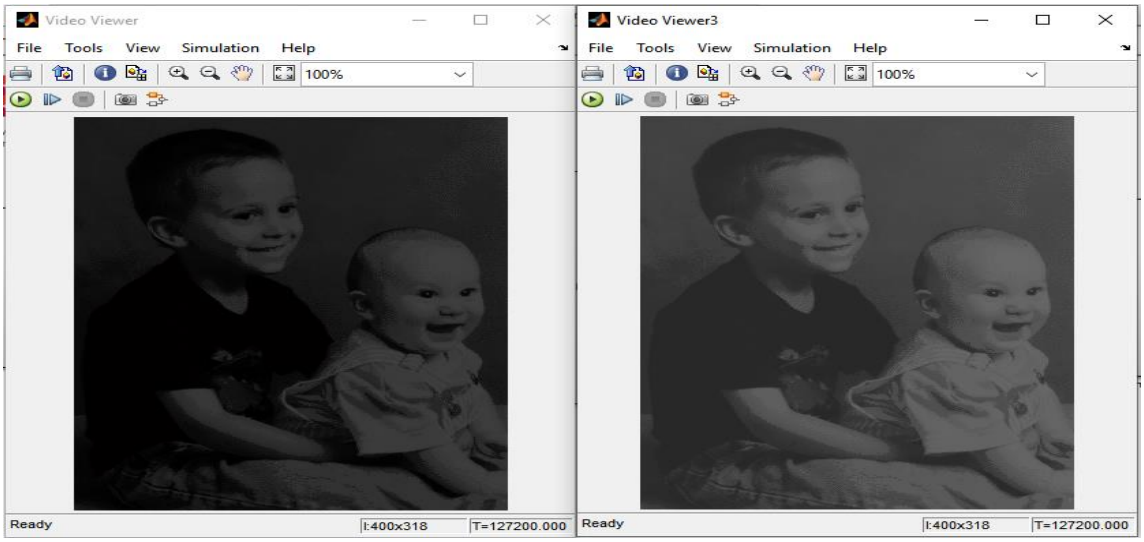


Figure. 2.21. Result for grayscale image enhancement

A similar approach is used to resolve multidimensional R|G|B image signals for color images.

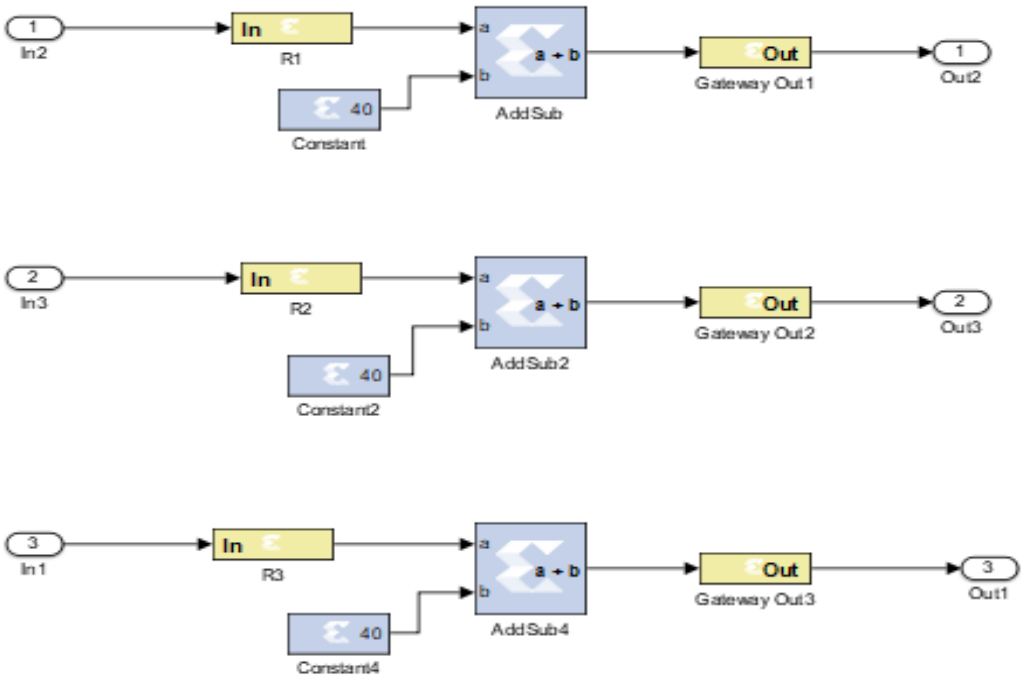


Figure. 2.22. XSG design of color image enhancemnet

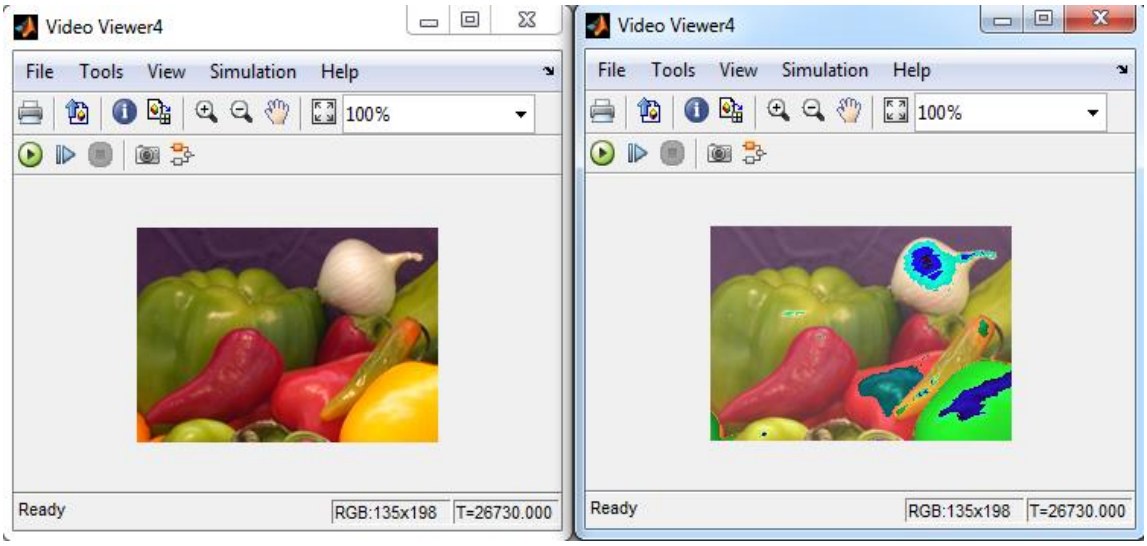
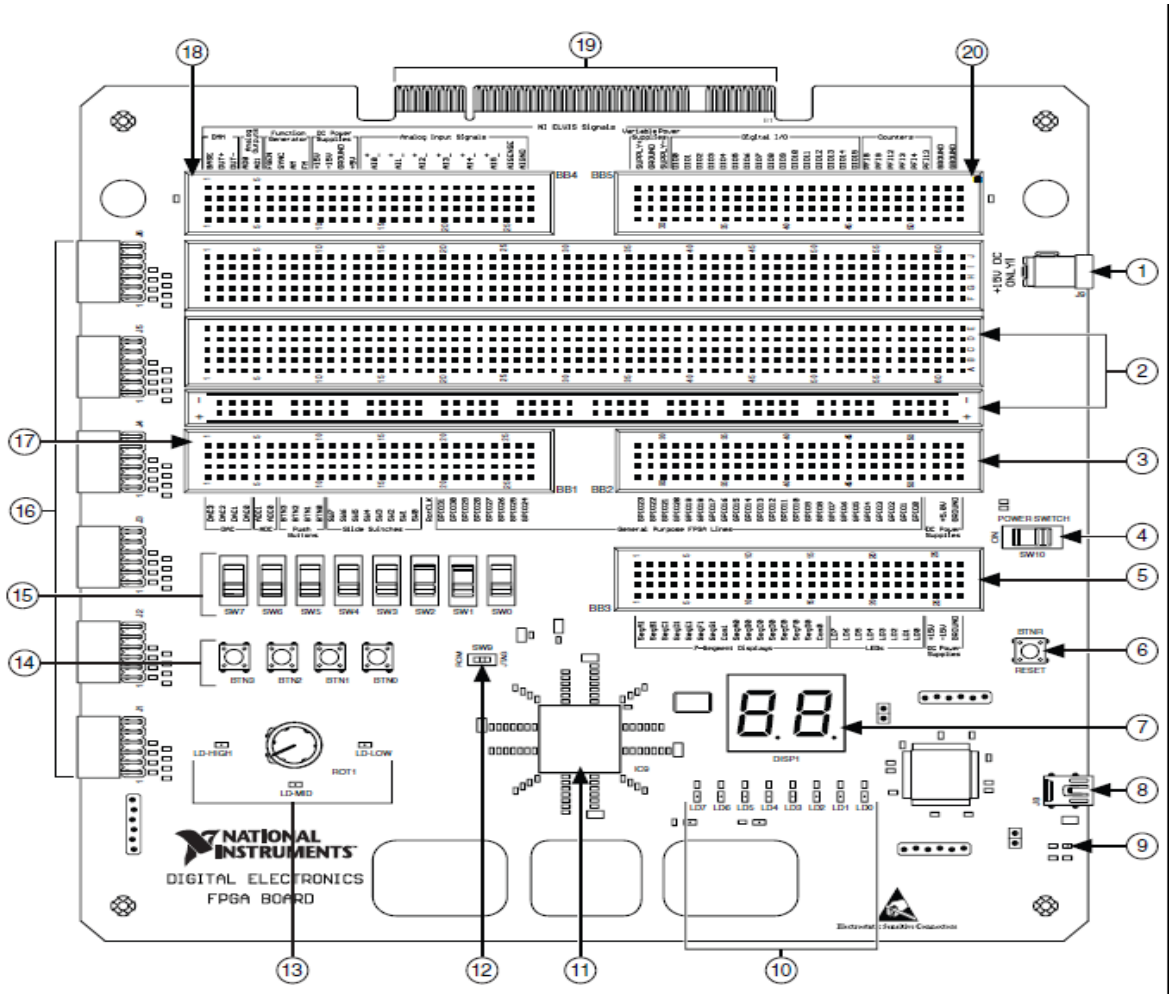


Figure. 2.23. Result for color image enhancement

2.6 NI Digital Electronics FPGA Board Hardware Platform

The hardware platform used in this project is NI Digital Electronics FPGA Boardest a circuit development platform based on the FPGA XC3S500E Xilinx Spartan-3E. The figure below shows the reference diagram of the top view of the NI Digital Electronics FPGA board.



1	Power Connector	7	Seven-Segment Displays	14	Push Buttons
2	General-Purpose Breadboard Banks	8	USB Connector	15	Slide Switches
		9	LD-G LED	16	Digilent Pmod Connectors
3	Signal Breadboard Bank BB2	10	LEDs	17	Signal Breadboard Bank BB1
4	Power Switch	11	FPGA	18	Signal Breadboard Bank BB4
5	Signal Breadboard Bank BB3	12	Switch SW9	19	NI ELVIS Connector
6	Reset Button	13	Rotary Push-Button Knob/LEDs	20	Signal Breadboard Bank BB5

Figure. 2.24. NI Digital Electronics FPGA Board.

2.7 FPGA XC3S500E Xilinx Spartan-3E

The FPGA XC3S500E Xilinx Spartan-3E is a Plastic Leaded Chip Carrier (PLCC) type integrated circuit with the following case marking:

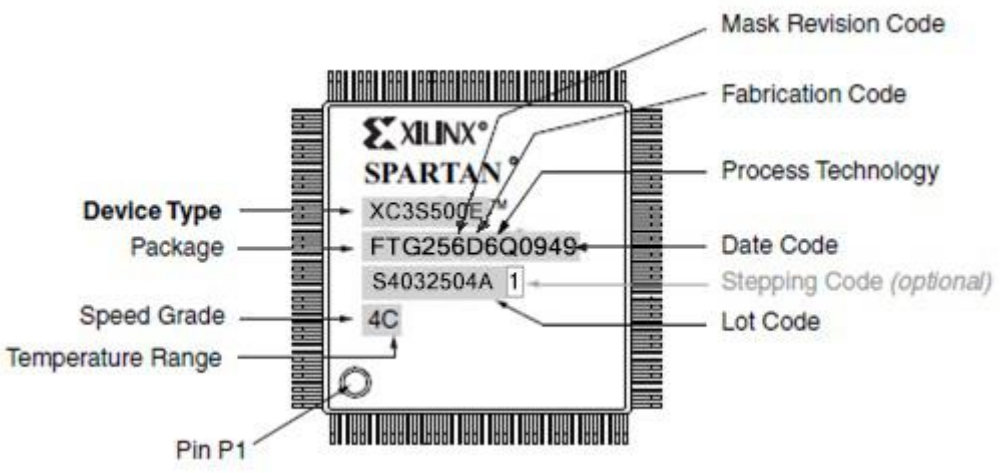
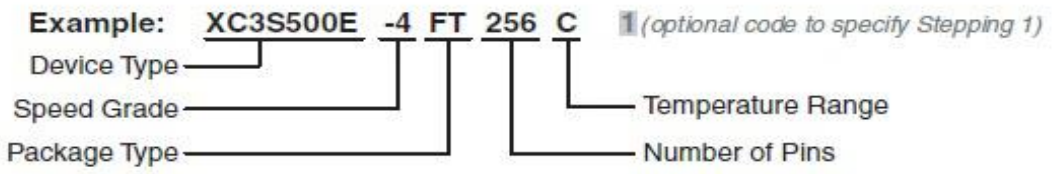


Figure. 2.25. FPGA XC3S500E

The marking information is:



2.8 Hardware co-Simulation

The complete module should be transformed to an FPGA synthesizable one in order to implement this design in an FPGA board. The core module for any image processing is adapted for JTAG hardware co-simulation with the help of the System generator token. A new window will popup when you click the system generator token, as shown in Figure 2.26. After this block is configured for the target platform Digital Electronics FPGA and the Spartan 3E (XC3S500E-4FT256) is used for board level implementation in this project. A hardware co-simulation block will be constructed after pressing the generate button in the System generator block.

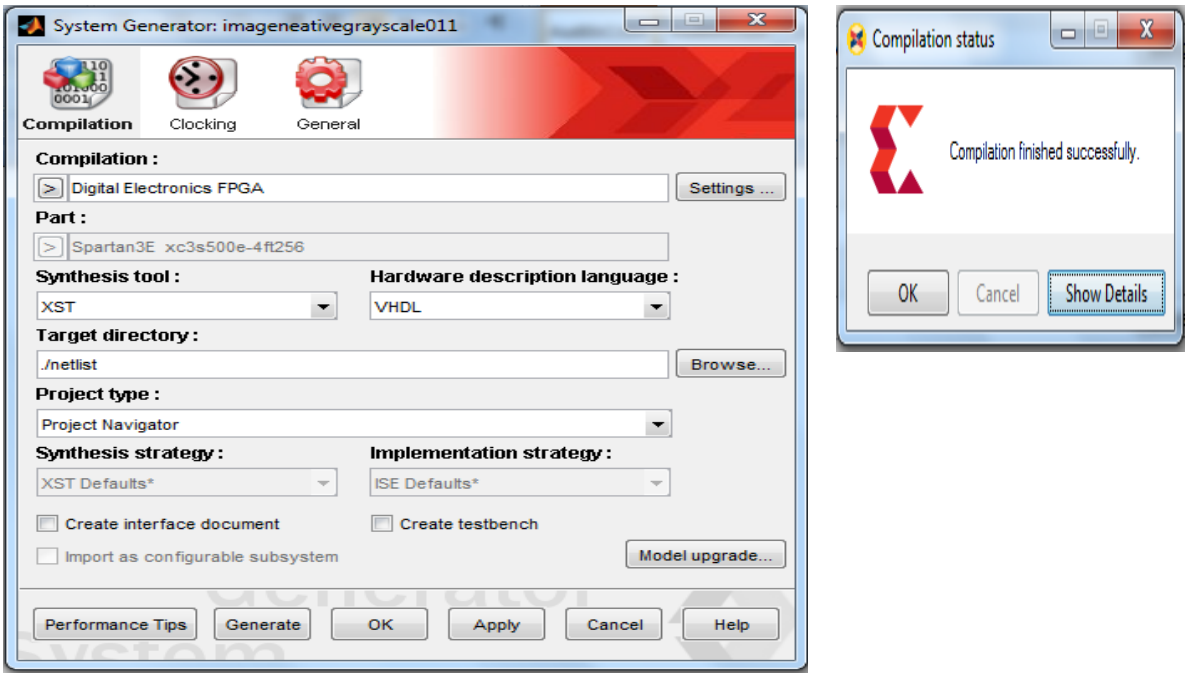


Figure. 2.26. Compilation co-simulation and code generation

The module for image negative is converted to JTAG hardware co-simulation.

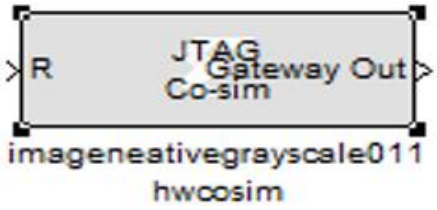


Figure. 2.27. Hardware Co-simulation block for grayscale image negative

The hardware co-simulation block was added to the design to do the hardware software co-simulation, and we can now see FPGA and XSG/software output at the same time. Figure 3.9 depicts the whole architecture for Blurring/Filtering, including the hardware and software co-simulation design.

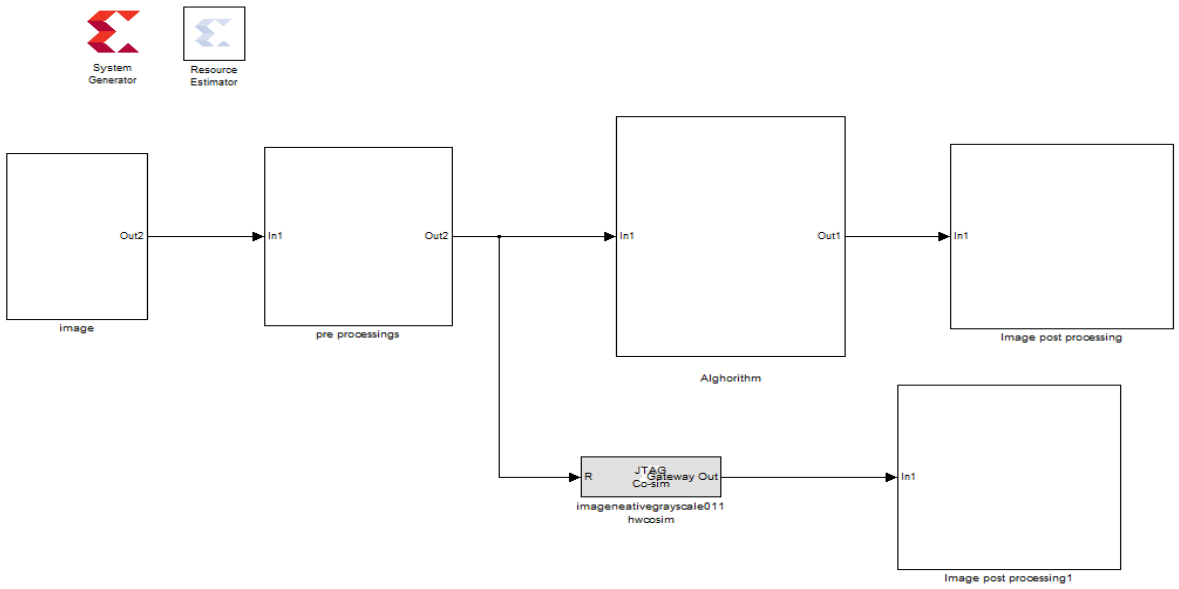


Figure. 2.28. Hardware software Co-simulation for the grayscale image negative

2.9 Hardware result

When the system design is simulated in Simulink, the compiled component of the result is calculated in actual FPGA hardware, which typically results in a substantially faster simulation time while proving the hardware's functional correctness. Hardware implementation of grayscale image negative for “cameraman.tif” is tabulated below.

Figure. 2.29. Hardware result of grayscale image negative using JTAG Co-simulation block

2.10 Conlusion

In this chapter, we modeled the basic image processing algorithms with the XSG tool and then simulated the model and subsequently transferred the design to FPGA using Xilinx's Hardware Co-Simulation. Finally, the algorithms were validated on the Digital Electronics FPGA Board (DEFB) development platform. we conclude that the Xilinx system generator is an incredibly useful tool for programming and hardware image processing tasks.

CONCLUSION

The purpose of this study is to implement the basic image processing algorithms on an FPGA platform using a compilation for hardware co-simulation.

We have modeled and simulated the image processing algorithms starting with grayscale and color image negative, contrast stretching, enhancement and image thresholding using the powerful tool Xilinx System Generator XSG. This tool offers the possibility to use the high-level abstraction language of MATLAB and the graphical blocks of Simulink and Xilinx simultaneously. Indeed, the latter also provides a hardware Co-Simulation allowing the direct incorporation of the design running in an FPGA in a Simulink simulation as well as the transfer of the design to the FPGA.

In conclusion, we can say that the Xilinx System Generator provides simplicity and ease for hardware implementation and offers fast means for hardware implementation of complex techniques used for image processing.

References

[1] KHADRAOUI-MESMARI-HANED -Mémoire de MASTER -UNIVERSITE KASDI MERBAH OUARGLA

[2] Polycopié de cours 1 Université Ferhat Abbas — Sétif

[3] abstract of 2010 Second International Conference on Computer Engineering and Applications < Color Image to Grayscale Image Conversion >

[4] S. R. A. S. J.C.Moctezuma, "Architecture for filtering images using Xilinx System Generator," World Scientific Advanced Series In Electrical And Computer Engineering, Proceeding of the 2nd WSEAS International Conference on Computer Engineering and Applications, pp. 284-289, 2008.

[5] M. Ownby and M. W.H, "A Design Methodology for Implementing DSP with Xilinx System Generator for Matlab," IEEE International Symposium on System Theory., pp. 404-408, 2003.

[6] Matlab guide de l'utilisateur

[7] Kumar, K. A., & Kumar, M. V. (2014). Implementation of image processing lab using Xilinx System Generator. Advances in Image and Video Processing, 2(5), 27-35.

[8] Durgakeri, B. S., & Chiranjeevi, G. N. (2019, May). Implementing Image Processing Algorithms using Xilinx System Generator with Real Time Constraints. In 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) (pp. 230-234). IEEE.

[9] A. K. Jain, “Fundamentals of digital image processing,” Englewood Cliffs, NJ: Prentice Hall, 1989.

[10] S. Martins and J. C. Alves, “A high-level tool for the design of custom image processing systems,”

[11] Suthar, A. C., Vayada, M., Patel, C. B., & Kulkarni, G. R. (2012). Hardware software co-simulation for image processing applications. International Journal of Computer Science Issues (IJCSI), 9(2), 560.

[12] System Generator for DSP Reference Guide UG638 (v14.5) March 20, 2013.

[12] Raut, N. P., & Gokhale, A. V. (2013). FPGA implementation for image processing algorithms using xilinx system generator. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), 2(4), 26-36.

[14] Dakre, K. A., & Pusdekar, P. N. (2015). Image enhancement using hardware co-simulation for biomedical applications. International Journal on Recent and Innovation Trends in Computing and Communication, 3(2), 869-877.